DW_pulseack_sync

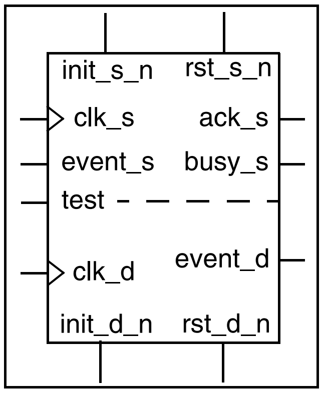
Pulse Synchronizer with Acknowledge

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Fully tested cross-clock domain
- Fully parameterized
- Able to use both positive and negative clock edge for sending clock domain
- Provides for both combinatorial and registered output, via parameter



Description

DW_pulseack_sync provides a low-risk method for transmitting single-clock-cycle pulses between two different clock domains. This component uses clock-domain-crossing techniques to safely transfer pulses between logic operating on different clocks, as well as providing a busy signal and acknowledge to guarantee the pulse has arrived in the destination domain.

Simulation models are available in Verilog and VHDL. Synthesizable source is available in Verilog.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk_s	1	Input	Source clock
rst_s_n	1	Input	Asynchronous source reset
init_s_n	1	Input	Synchronous source reset
event_s	1	Input	Input pulse
busy_s	1	Output	`busy_s` is active high while the transmitted pulse and the acknowledge are in transit
ack_s	1	Output	`ack_s` is active high when the transmitted pulse has traveled to the destination domain and returned to the transmitting domain; active for one clock cycle
clk_d	1	Input	Destination clock
rst_d_n	1	Input	Asynchronous destination reset
init_d_n	1	Input	Synchronous destination reset
event_d	1	Output	Output pulse
test	1	Input	Scan test mode select input

Table 1-2 Parameter Description

Parameter	Values	Description
reg_event	0 to 1 Default: 1	■ 0: No register on output ■ 1: Register event_d output
reg_ack	0 to 1 Default: 1	■ 0: `ack_s` has combination logic, but latency is 1 cycle sooner ■ 1: `ack_s` is retimed to eliminate combinational logic in the output; requires an additional clock cycle of delay
ack_delay	0 to 1 Default: 1	■ 0: `ack_s` has combination logic, but latency is 1 cycle sooner ■ 1: `ack_s` is retimed so there is no logic between register and port, but event is delayed 1 cycle
f_sync_type	0 to 4 Default: 2	■ 0: Single clock design `clk_d=clk_s` ■ 1: Negedge to posedge sync ■ 2: Posedge to posedge sync ■ 3: 3 posedge registers in destination domain ■ 4: 4 posedge registers in destination domain
r_sync_type	0 to 4 Default: 2	■ 0: Single clock design `clk_d=clk_s` ■ 1: Negedge to posedge sync ■ 2: Posedge to posedge sync ■ 3: 3 posedge registers in destination domain ■ 4: 4 posedge registers in destination domain
tst_mode	0 to 2 Default: 0	■ 0: No test latch insertion ■ 1: Hold latch using negedge flop ■ 2: Hold latch using active low latch
verif_en	0 to 4 Default: 1	■ 0: No sampling errors inserted ■ 1: Sampling errors are randomly inserted with 0 or up to 1 destination clock cycle delays ■ 2: Sampling errors are randomly inserted with 0, 0.5, 1, or 1.5 destination clock cycle delays ■ 3: Sampling errors are randomly inserted with 0, 1, 2, or 3 destination clock cycle delays ■ 4: Sampling errors are randomly inserted with 0 or up to 0.5 destination clock cycle delays

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
pulse_mode	0 to 3 Default: 0	Selects the type of pulse presented to the input. ■ 0: Single source domain clock cycle pulse transmitted to destination domain ■ 1: Rising transition detect transmitted as single cycle pulse in the destination domain ■ 2: Falling transition detect transmitted as single clock cycle pulse in the destination domain ■ 3: Toggle operation (rising or falling) is transmitted as a single-cycle pulse in the destination domain

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

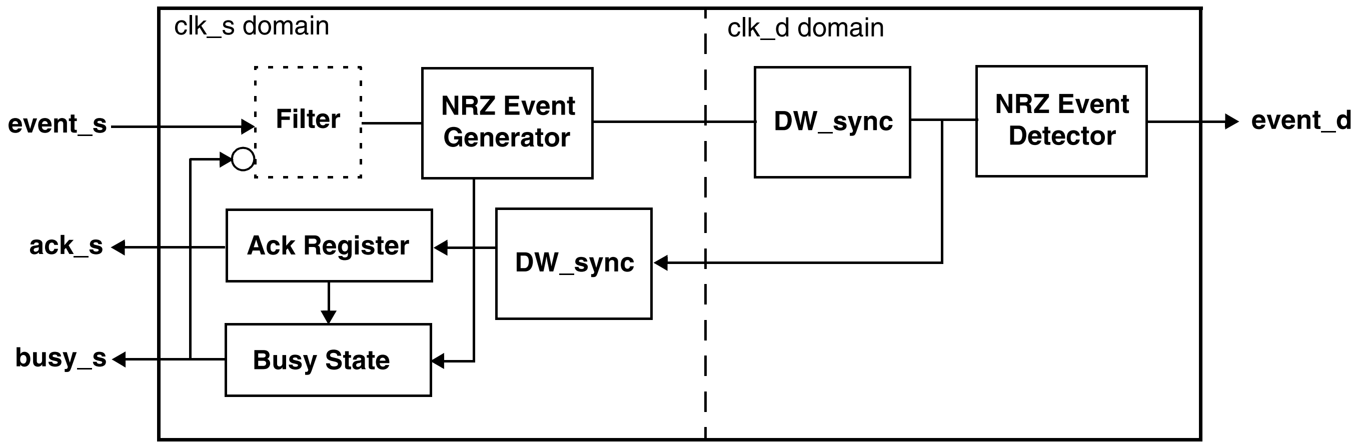
Table 1-4 Simulation Models

Model	Function
DW03.DW_PULSEACK_SYNC_SIM	Architectural name for VHDL simulation without missampling.
DW03.DW_PULSEACK_SYNC_SIM_MS	Architectural name for VHDL simulation with missampling enabled; see also "Simulation Methodology" below
dw/dw03/src/DW_pulseack_sync_sim.vhd	VHDL simulation model source code (modeling RTL)
dw/sim_ver/DW_pulseack_sync_1c.v	Verilog simulation model source code

Functional Description

The DW_pulseack_sync synchronizer provides a method of passing event information from one clock domain to another and acknowledgment that the pulse was recognized in the destination domain. Using a Non-Return-to-Zero (NRZ) event generator (that is, a toggle register) triggered by the input, `event_s`, in the source clock domain, the destination domain synchronizes the NRZ event signal (using an instance of DW_sync) and then detects the NRZ event (that is, a toggle of the signal) and presents an active high output on `event_d` for once cycle of `clk_d` to signal the detected event.

Figure 1-1 DW_pulse_sync Dual-clock Pulse Synchronizer Block Diagram



The way in which the `event_s` input triggers events depends on the value of the parameter, *pulse_mode*.

Table 1-5 *pulse_mode* Parameter Trigger Method

pulse_mode	Trigger Method
0	Trigger an event for each clock cycle that `event_s` is high
1	Trigger an event for each clock cycle when `event_s` is high AND `event_s` was low for the previous clock (rising edge detect)
2	Trigger an event for each clock cycle when `event_s` is low AND `event_s` was high for the previous clock (falling edge detect)
3	Trigger an event for each clock cycle when `event_s` is different than it was for the previous clock (rising- or falling-edge detect)

Although there is no required clock frequency relationship between the source and destination clocks, the rate of events that the synchronizer can reliably pass is restricted by the frequency of the destination clock. The time between NRZ events between the domains must not approach one clock period of `clk_d` plus some adequate setup time of a synchronization register. To be safe, allow at least two periods of `clk_d` between any two consecutive events.

To avoid unwanted events, both domains should be reset.

Simulation Methodology

Because this component contains synchronizing devices, there are two methods available for simulation. One method is to use the simulation models that emulate the RTL model. Or, you can enable modeling of random skew between bits of signals traversing to and from each domain (called missampling).

To use the simulation models that emulate the RTL model, no special configuration is required.

To use missampling requires the following considerations:

- To enable missampling in Verilog simulations, define the macro DW_MODEL_MISSAMPLES:

  ```
  `define DW_MODEL_MISSAMPLES
  ```

  If `DW_MODEL_MISSAMPLES is defined, the *verif_en* parameter comes into play to configure the simulation model as described by Table 1-2 on page 2. If `DW_MODEL_MISSAMPLES is not defined, the Verilog simulation model behaves as if *verif_en* is set to 0.

- To enable missampling in VHDL simulations, a simulation architecture named sim_ms is provided. The parameter *verif_en* only has meaning when using sim_ms. That is, when the `sim` simulation architecture is used instead, the model behaves as though *verif_en* is set to 0.

Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If values other than 1 or 0 are present on a clock port, the following message is displayed:

  ```
  WARNING: <instance_path>.<clock_name>_monitor:
      at time = <timestamp>, Detected unknown value, x, on <clock_name> input.
  ```

  To suppress only this warning message for all DWBB components, use the following macro:

  - Define the DW_DISABLE_CLK_MONITOR macro. You can define this macro in the following ways:
    - Specify the Verilog preprocessing macro in Verilog code:

      ```
      `define DW_DISABLE_CLK_MONITOR
      ```

    - Or, include a command line option to the simulator, such as:

      `+define+DW_DISABLE_CLK_MONITOR` (which is used for the Synopsys VCS simulator)

  This message is also suppressed using the DW_SUPPRESS_WARN macro explained earlier.

Related Topics

- Memory – Registers Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE,WORK;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;

entity DW_pulseack_sync_inst is
      generic (
        inst_reg_event : NATURAL := 1;
        inst_reg_ack : NATURAL := 1;
        inst_ack_delay : NATURAL := 0;
        inst_f_sync_type : NATURAL := 2;
        inst_r_sync_type : NATURAL := 2;
        inst_tst_mode : NATURAL := 0;
        inst_verif_en : NATURAL := 1;
        inst_pulse_mode : NATURAL := 0
        );
      port (
        inst_clk_s : in std_logic;
        inst_rst_s_n : in std_logic;
        inst_init_s_n : in std_logic;
        inst_event_s : in std_logic;
        inst_clk_d : in std_logic;
        inst_rst_d_n : in std_logic;
        inst_init_d_n : in std_logic;
        inst_test : in std_logic;
        busy_s_inst : out std_logic;
        ack_s_inst : out std_logic;
        event_d_inst : out std_logic
        );
    end DW_pulseack_sync_inst;


architecture inst of DW_pulseack_sync_inst is

begin

    -- Instance of DW_pulseack_sync
    U1 : DW_pulseack_sync
    generic map ( reg_event => inst_reg_event,
                  reg_ack => inst_reg_ack,
                ack_delay => inst_ack_delay,
                f_sync_type => inst_f_sync_type,
                r_sync_type => inst_r_sync_type,
                tst_mode => inst_tst_mode,
                verif_en => inst_verif_en,
                pulse_mode => inst_pulse_mode )
    port map ( clk_s => inst_clk_s,
               rst_s_n => inst_rst_s_n,
             init_s_n => inst_init_s_n,
             event_s => inst_event_s,
             clk_d => inst_clk_d,
             rst_d_n => inst_rst_d_n,
             init_d_n => inst_init_d_n,
             test => inst_test,
             busy_s => busy_s_inst,
             ack_s => ack_s_inst,
             event_d => event_d_inst );


end inst;
-- pragma translate_off
library DW03;
configuration DW_pulseack_sync_inst_cfg_inst of DW_pulseack_sync_inst is
  for inst
  end for; -- inst
end DW_pulseack_sync_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_pulseack_sync_inst( inst_clk_s, inst_rst_s_n, inst_init_s_n, inst_event_s,
inst_clk_d,
          inst_rst_d_n, inst_init_d_n, inst_test, busy_s_inst, ack_s_inst,
          event_d_inst );

parameter reg_event = 1;
parameter reg_ack = 1;
parameter ack_delay = 0;
parameter f_sync_type = 2;
parameter r_sync_type = 2;
parameter tst_mode = 0;
parameter verif_en = 1;
parameter pulse_mode = 0;


input inst_clk_s;
input inst_rst_s_n;
input inst_init_s_n;
input inst_event_s;
input inst_clk_d;
input inst_rst_d_n;
input inst_init_d_n;
input inst_test;
output busy_s_inst;
output ack_s_inst;
output event_d_inst;

    // Instance of DW_pulseack_sync
    DW_pulseack_sync #(reg_event, reg_ack, ack_delay, f_sync_type, r_sync_type,
tst_mode, verif_en, pulse_mode)
      U1 ( .clk_s(inst_clk_s), .rst_s_n(inst_rst_s_n), .init_s_n(inst_init_s_n),
.event_s(inst_event_s), .clk_d(inst_clk_d), .rst_d_n(inst_rst_d_n),
.init_d_n(inst_init_d_n), .test(inst_test), .busy_s(busy_s_inst), .ack_s(ack_s_inst),
.event_d(event_d_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
September 2021	DWBB_202106.2	■ Added the description of the *pulse_mode* parameter in Table 1-2 on page 2 and updated the description in Table 1-5 on page 4
July 2020	DWBB_201912.5	■ Expanded the description of how to enable missampling during simulation in "Simulation Methodology" on page 5 ■ Adjusted content and title of "Suppressing Warning Messages During Verilog Simulation" on page 5 and added the DW_SUPPRESS_WARN macro
October 2019	DWBB_201903.5	■ Added the "Disabling Clock Monitor Messages" section ■ Added this Revision History table and the document links on this page